IA-PC HPET (High Precision Event Timers) Specification

Revision: 1.0
Date: June 2004

LEGAL DISCLAIMER

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Intel disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppels or otherwise, to any intellectual property rights is granted herein.

This specification is a preliminary draft provided for comment and informational purposes only, and is subject to change without any notice, obligation or liability. Readers should not rely on this specification in any way for product design purposes.

It is Intel's intent to provide a Version 1.0 of this specification, that will be made available subject to an appropriate license agreement. However, Intel is under no obligation or liability to do so.

The contents of the areas marked as "reserved" in this specification will not be licensed under the Intel license for this specification.

IA-PC HPET Specification
Copyright © 1999-2004 Intel Corporation
All rights reserved.

*THIRD-PARTY BRANDS AND NAMES MAY BE CLAIMED AS THE PROPERTY OF OTHERS.

Table of Contents

1. IA-PC HPET

1.1 Revision History:

Version	Comments
0.97	Last Updated: 03/07/2000 • Incorporated technical editing changes, released for external feedback.
0.97a	Last Updated: 05/18/2000 • Incorporated various non-technical and legal feedbacks.
0.98	01/20/2002 • Product name changed: from **Multimedia Timer** to **HPET** (High Precision Event Timer) • Technologic term changed: from **Legacy Mode** to **LegacyReplacement Mode** for clarity purpose • ETDT ACPI table changed: **ETDT** (Event Timer Descriptor Table) is changed to **HPET** table and its content of the table has been updated. • IA64 platform support: Use GAS(Generic Address Structure) format in HPET table and up to 64KB timer block.
0.98a	08/31/2001 • Modified the accuracy of clock frequency drift to 0.05% • Add “write lock” note to the programming requirement
1.0	6/8/2004 • Removed color-code for read-only fields • Added programming notes for 64-bit register access in a 64-bit platform • Explicitly mark “Reserved” in reserved fields of FSB Registers

1.2 Scope

This specification provides register model and programming interface definitions for new event timer hardware for use on Intel Architecture-based Personal Computers. In this specification, the terms 'IA-PC HPET and 'Event Timers' refer to the same timer hardware.

The IA-PC HPET Specification defines timer hardware that is intended to initially supplement and eventually replace the legacy 8254 Programmable Interval Timer and the Real Time Clock Periodic Interrupt generation functions that are currently used as the 'de-facto' timer hardware for IA-PCs.

This new timer hardware can be used by system software for:

- **Synchronizing**
  - Real-Time Digital Audio & Video Streams
    - 64-bit free running up-counter
- **Scheduling**
  - Threads, Tasks, Processes, etc.
    - Fixed Rate (Periodic) Interrupt Generation
      - System Heart Beat
      - Non-Real Time Thread Scheduler
    - Variable Rate (One-Shot) Interrupt Generation
      - Scheduling real time tasks associated with host-based signal processing applications
- **Time Stamping**
  - On Multiprocessor platforms
    - 64-Bit free running up-counter can be utilized as DIG64 "platform timer" for Time Stamping Applications. This provides a time-base that is insensitive to clock frequency drifts on individual CPU's on a N-Way MP systems.

Note:
The name of the timer block has been changed from Multimedia Timer to HPET (High Precision Event Timer). However, before the new name was adopted, many related documents continue to use or reference the term of "Multimedia Timer". Therefore, for the purposes of designing products to this specification, the terms HPET, Multimedia Timer, MMT and MM Timer should be treated as the same timer hardware.

1.3 Terminology

Term	Definition for this document
IA	Intel Architecture
PC	Personal Computer
IA-PC	Intel Architecture-based PC
PIT	8254 Programmable Interval Timer
RTC	Real Time Clock
SCI	System Configuration Interrupt
FSB	Front Side Bus
MM	Multimedia
• Timer • Event Timer • HPET • MM Timer • MMT	The terms Timer, Event Timer, HPET, MMT and MM Timer refer to the combination of a Counter, Comparator, and Match Register. The Comparator compares the contents of the Match Register against the value of a free running up-counter. When the output of the up-counter equals the value in the match register an interrupt is generated. The IA-PC HPET Architecture allows up to 32 compare/match registers per counter. Each of the 32 comparators can output an interrupt.
Timer Block	Each Timer Block consists of a single counter that feeds up to 32 comparators. Each Timer Block in the system can have different clocking attributes.
32-Bit Timer	Comparator Register is 32 bits wide. Main Counter can be 32 or 64 bits wide for a ’32 bit Timer’.
64-Bit Timer	Comparator Register is 64 bits wide. Main Counter must be 64 bits wide for a ’64 bit Timer’.

Note: For better legibility, the fields of HPET internal registers are color-coded as following:

R	Purple background indicates reserved fields

<b1:b2>: Represents bit filed from bit b1 to bit b2
<b1>: Represents a single bit field

2. Hardware Overview

The IA-PC HPET Architecture defines a set of timers that can be used by the operating system. The timers are defined such that in the future, the OS may be able to assign specific timers to be used directly by specific applications. Each timer can be configured to generate a separate interrupt. This specification allows for a block of 32 timers, with support for up to 8 blocks, for a total of 256 timers. However, specific implementations can include only a subset of these timers.

The timers are implemented as a single up-counter with a set of comparators. The counter increases monotonically. When software does two consecutive reads of the counter, the second read will never return a value that is less than the first read unless the counter has actually rolled over. Each timer includes a match register and a comparator. Each individual timer can generate an interrupt when the value in its match register equals the value of the free-running counter. Some of the timers can be enabled to generate a periodic interrupt.

The registers associated with these timers are mapped to memory space (much like the I/O APIC). However, it is not implemented as a standard PCI function. The BIOS reports to the operating system the location of the memory-mapped register space consumed by the timers. The hardware can support re-locatable address decode space, however the BIOS will set this space prior to handing it over to the OS. It is not expected that the OS will move the location of these timers once it is set by the BIOS.

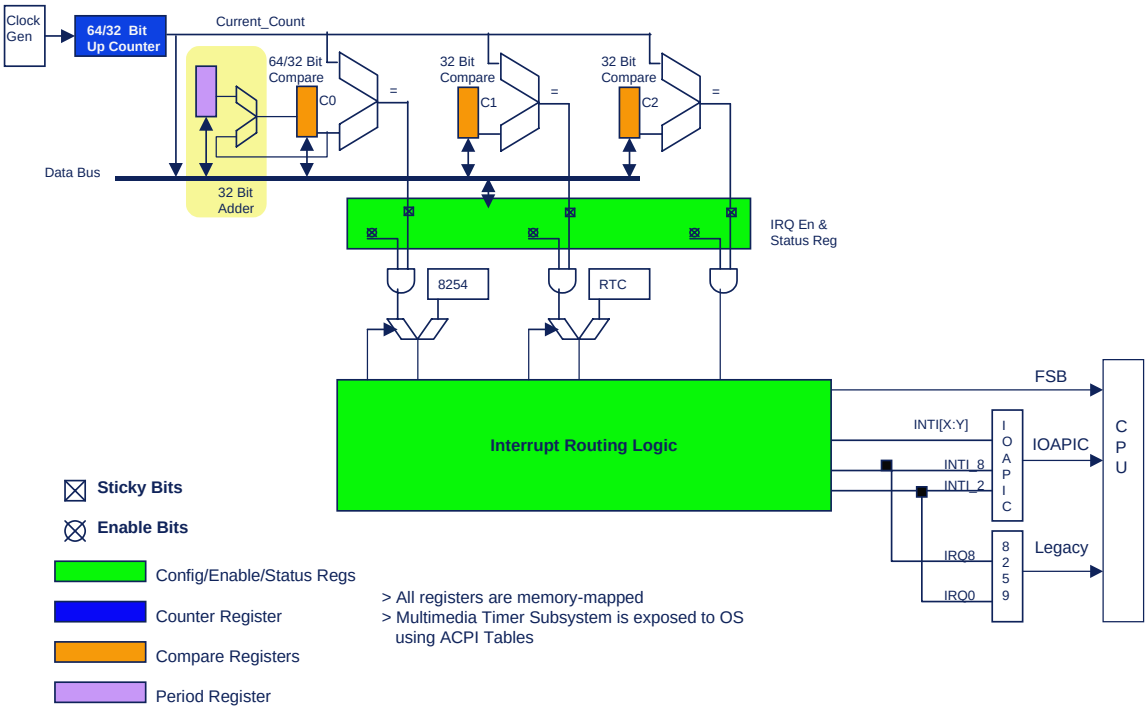


Figure 1 Hardware Block Diagram

2.1 Register Model Overview

2.1.1 Memory Map

The Event Timer registers are memory mapped in a non-indexed scheme. This allows the CPU to directly access each register without having to use an index register. The timer register space is 1024 bytes. The registers are generally aligned on 64-bit boundaries to simplify implementation with IA64 processors. For IA64 platform, the timer register space can be up to 64K bytes with page protection capability. The register model allows each timer block to contain up to 32 timers, where each 'timer' consists of a comparator plus a match register.

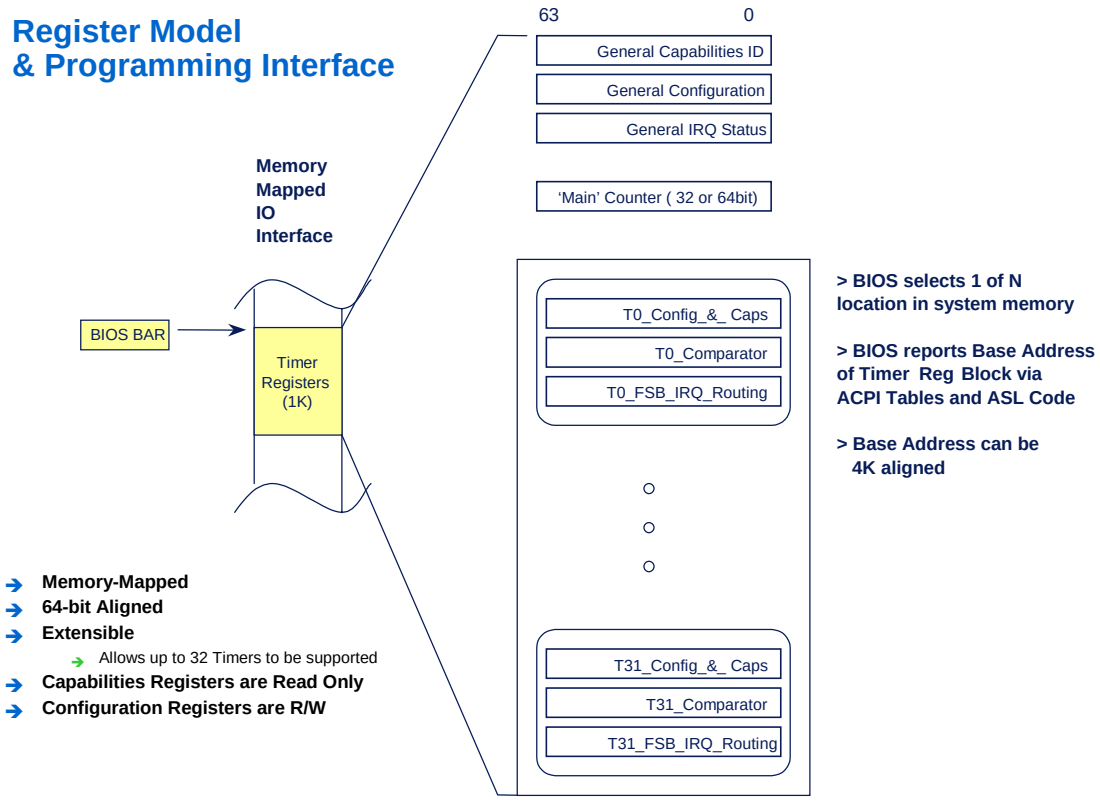


Figure 2 Register Model Overview

2.2 Minimum Recommended Hardware Implementation

Item	Recommendation	Comments
Main Counter Main Counter Width	Required to be an up-counter 64-bits	
Clock Frequency Clock Frequency Drift	Fmin = 10 MHz +- .05 % (500 ppm) +- .2% (200 ppm)	 Over any interval >= 1 Millisecond Over any interval <= 100 Microseconds
Number of Comparators Width of Comparators	3 32 bits (Minimum)	 If 64 Bits, must have 32-bit mode for IA32 platforms
Number of Periodic Capable Timers Width of Adder on Periodic Capable Timers	1 of 3 32 bits (Minimum)	
Number of One-shot Capable Timers	All 3	
Interrupt Delivery via 8259	Optional	LegacyReplacement IRQ Routing required for systems that intend to replace/supplement 8254/RTC legacy timers with this new timer architecture.
Interrupt Delivery via IOxAPIC	Required	LegacyReplacement IRQ Routing required for systems that intend to replace/supplement 8254/RTC legacy timers with this new timer architecture.
Interrupt Delivery via CPU FSB	Optional	

Table 1 Minimum Recommended Hardware Implementation

2.3 Register Definitions

2.3.1 Register Overview

Offset	Register	Type
000-007h	**General Capabilities and ID Register**	Read Only
008-00Fh	Reserved	
010-017h	**General Configuration Register**	Read-Write
018-01Fh	Reserved	
020-027h	**General Interrupt Status Register**	Read/Write Clear
028-0EFh	Reserved	
0F0-0F7h	**Main Counter Value Register**	Read/Write
0F8-0FFh	Reserved	
100-107h	**Timer 0 Configuration and Capability Register**	Read/Write
108-10Fh	**Timer 0 Comparator Value Register**	Read/Write
110-117h	**Timer 0 FSB Interrupt Route Register**	Read/Write
118-11Fh	Reserved	
120-127h	**Timer 1 Configuration and Capability Register**	Read/Write
128-12Fh	**Timer 1 Comparator Value Register**	Read/Write
130-137h	**Timer 1 FSB Interrupt Route Register**	Read/Write
138-13Fh	Reserved	
140-147h	**Timer 2 Configuration and Capability Register**	Read/Write
148-14Fh	**Timer 2 Comparator Value Register**	Read/Write
150-157h	**Timer 2 FSB Interrupt Route Register**	Read/Write
158-15Fh	Reserved	
160-3FFh	Reserved **for Timers 3-31**	

Table 2 Memory-Mapped Registers

2.3.2 Programming Requirements

1. Software must not attempt to read or write across register boundaries. For example, a 32-bit access should be to offset 00h, 04h, 08h, or 0Ch. 32-bit accesses should not be to 01h, 02h, 03h, 05h, 06h, 07h, 09h, 0Ah, 0Bh, 0Dh, 0Eh, or 0Fh. 64-bit accesses can only be to 00h and must not cross 64-bit boundaries.
2. Software should not write to read-only registers.
3. Software should not expect any particular or consistent value when reading reserved registers or bits.
4. Software should perform read-modify-write operations on reserved bits.

> **Note:**
> **Host controllers are not required to support exclusive-access mechanisms (such as PCI LOCK) for accesses to the memory-mapped register space. Therefore, if software attempts exclusive-access mechanisms to the host controller memory-mapped register space, the results are undefined.**

2.3.3 Power Management Considerations

It is the Operating System's responsibility to save and restore Event Timer hardware context if this needs to be preserved through ACPI System Sleep State transitions.

General behavioral rules for Event Timer hardware regarding sleep state transitions:

1. The Event Timer registers (including the main counter) are not expected to be preserved through an S3, S4, or S5 state.

2. The features and functions associated with these registers are not expected to be used in an S1 state. Prior to going to an S1 state, all interrupts associated with this function should be disabled.
3. The main counter is permitted, but not required to run during S1 or S2 states. This allows mobile systems to stop clock generators feeding the main counter during S1 or S2 states.

2.3.4 General Capabilities and ID Register

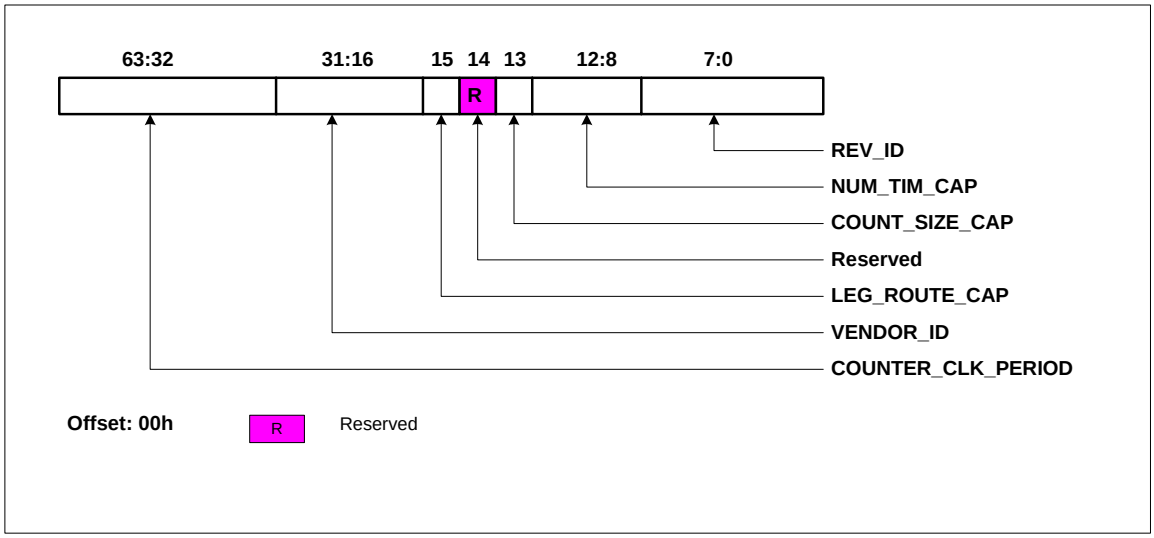


Figure 3 General Capability and ID Register

General Capability and ID Register Addressing

Offset	0x00
Attribute	Read-Only
Size	64-bits
General Behavioral Rules	1. Writes to this register should not be attempted by software. 2. Software can read the various bytes in this register using 32-bit or 64-bit accesses. 32-bit accesses can be done to offset 00h or 04h, but not to offsets 01h, 02h, 03h, 05h, 06h, or 07h. 64-bit accesses can only be done to 00h.

General Capabilities and ID Register Bit Definitions

Bit		Description
63:32	**COUNTER_CLK_PERIOD**	**Main Counter Tick Period:** This read-only field indicates the period at which the counter increments in femptoseconds (10^-15 seconds). A value of 0 in this field is not permitted. The value in this field must be less than or equal to 05F5E100h (10^8 femptoseconds = 100 nanoseconds). The resolution must be in femptoseconds (rather than picoseconds) in order to achieve a resolution of 50 ppm.
31:16	**VENDOR_ID**	This read-only field will be the same as what would be assigned if this logic was a PCI function.
15	**LEG_RT_CAP**	**LegacyReplacement Route Capable:** If this bit is a 1, it indicates that the hardware supports the LegacyReplacement Interrupt Route option.
14	**Reserved:**	In order to preserve usage of these bits in the future, software should always write a 0 to these bits until they are defined.
13	**COUNT_SIZE_CAP**	**Counter Size:** • This bit is a 0 to indicate that the main counter is 32 bits wide (and cannot operate in 64-bit mode). • This bit is a 1 to indicate that the main counter is 64 bits wide (although this does not preclude it from being operated in a 32-bit mode).
12:8	**NUM_TIM_CAP**	**Number of Timers:** This indicates the number of timers in this block. The number in this field indicates the last timer (i.e. if there are three timers, the value will be 02h, four timers will be 03h, five timers will be 04h, etc.).

7:0	**REV_ID**	This indicates which revision of the function is implemented. The value must NOT be 00h.

2.3.5 General Configuration Register

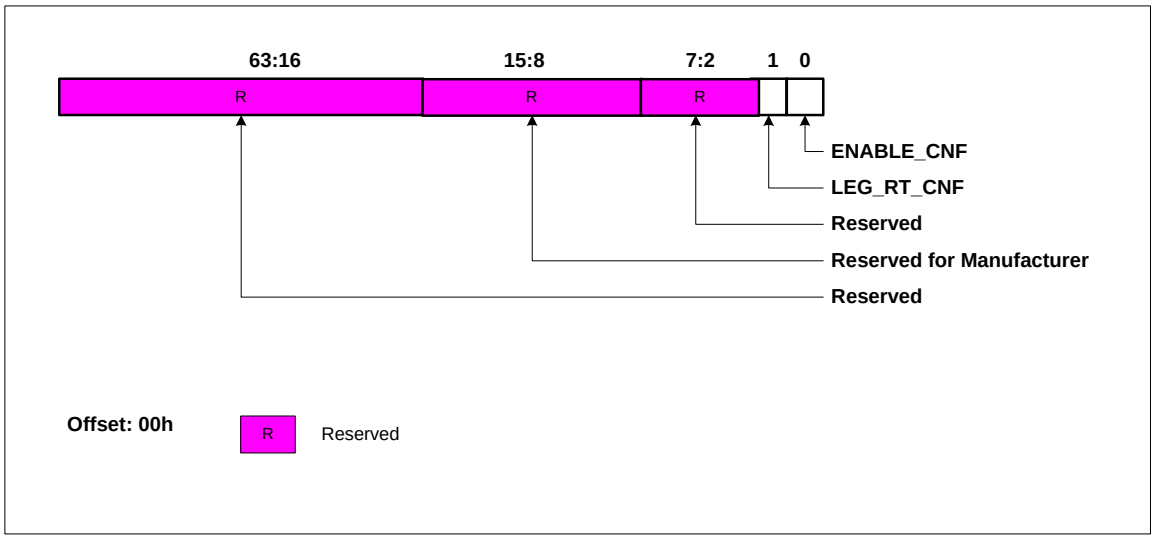


Figure 4 General Configuration Register

General Configuration Register Addressing

Offset	0x10
Attribute	Read-Write
Size	64-bits
General Behavioral Rules	1. Software can access the various bytes in this register using 32-bit or 64-bit accesses. 32-bit accesses can be done to offset 010h or 014h , but not to offsets 011h, 012h, 013h, 015h, 016h, or 017h. 64-bit accesses can only be done to 010h .

General Configuration Register Bit Definitions

Bit		Description
63:16	**Reserved**	In order to preserve usage of these bits in the future, software should not modify the value in these bits until they are defined. This is done by doing a "read-modify-write" to this register.
15:8	**Reserved for Non-OS**	These bits are reserved for the manufacturer. Future revisions of this spec will not use these bits. OS-based drivers must not modify the value in these bits. This is done by doing a "read-modify-write" to this register.
7:2	**Reserved:**	In order to preserve usage of these bits in the future, software should not modify the value in these bits until they are defined. This is done by doing a "read-modify-write" to this register.
1	**LEG_RT_CNF**	**LegacyReplacement Route:** • 0 – Doesn't support **LegacyRepalcement Route** • 1 – Supports **LegacyRepalcement Route** If the ENABLE_CNF bit and the LEG_RT_CNF bit are both set, then the interrupts will be routed as follows: Timer 0 will be routed to IRQ0 in Non-APIC or IRQ2 in the I/O APIC Timer 1 will be routed to IRQ8 in Non-APIC or IRQ8 in the I/O APIC Timer 2-n will be routed as per the routing in the timer n config registers.

		If the LegacyReplacement Route bit is set, the individual routing bits for timers 0 and 1 (APIC or FSB) will have no impact. If the LegacyReplacement Route bit is not set, the individual routing bits for each of the timers are used.
0	**ENABLE_CNF**	**Overall Enable:** This bit must be set to enable any of the timers to generate interrupts. If this bit is 0, then the main counter will halt (will not increment) and no interrupts will be caused by any of these timers. • 0 – Halt main count and disable all timer interrupts • 1 – allow main counter to run, and allow timer interrupts if enabled

2.3.6 General Interrupt Status Register

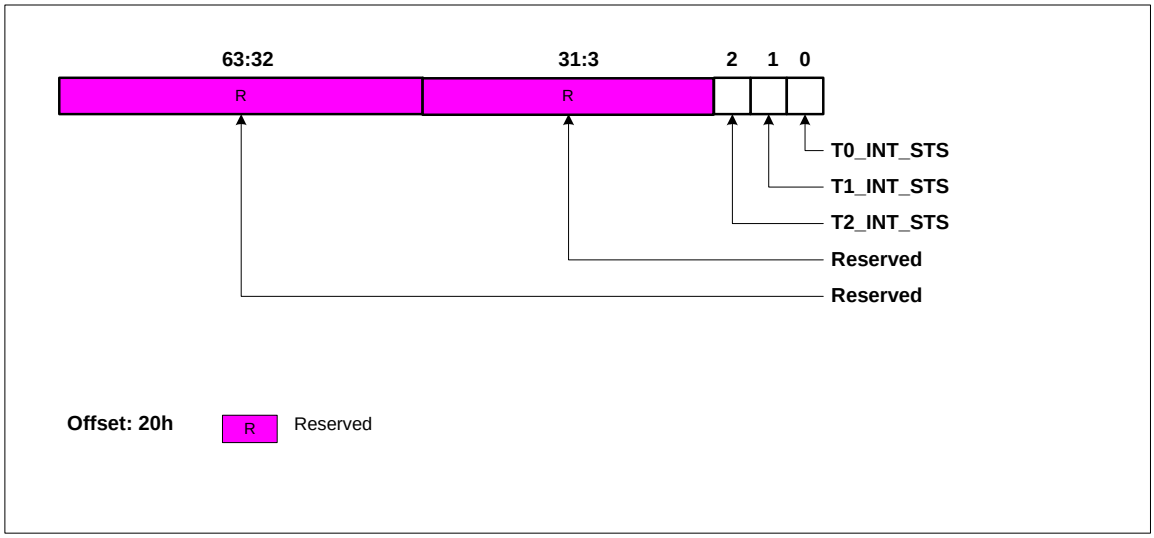


Figure 5 General Interrupt Status Register

General Interrupt Status Register Addressing

Offset	020h
Attribute	Read-Write Clear
Size	64-bits
General Behavioral Rules	1. Software can access the various bytes in this register using 32-bit or 64-bit accesses. 32-bit accesses can be done to offset 20h or 24h, but not to offsets 21h, 22h, 23h, 25h, 26h, or 27h. 64-bit accesses can only be done to 20h.

General Interrupt Status Register Field Definitions

Bit	Field Name	Description
63:32	**Reserved**	In order to preserve usage of these bits in the future, software should always write a 0 to these bits until they are defined.
31:3	**Tn_INT_STS**	**Timer n Interrupt Active (where xx is 31:3):** Same functionality as for Timer 0
2	**T2_INT_STS**	**Timer 2 Interrupt Active:** Same functionality as Timer 0.
1	**T1_INT_STS**	**Timer 1 Interrupt Active:** Same functionality as Timer 0.
0	**T0_INT_STS**	**Timer 0 Interrupt Active:** The functionality of this bit depends on whether the edge or level-triggered mode is used for this timer: **If set to level-triggered mode:** This bit defaults to 0. This bit will be set by hardware if the corresponding timer interrupt is active. Once the bit is set, it can be cleared by software writing a 1 to the same bit position. Writes of 0 to this bit will have no effect. For example if the bit is already set, a write of 0 will not clear the bit. **If set to edge-triggered mode:** This bit should be ignored by software. Software should always write 0 to this bit.

Note: Software uses Tn_INT_TYPE_CNF bit (bit <1> of Timer N Configuration and Capability Register) to select Level vs Edge operation.

2.3.7 Main Counter Register

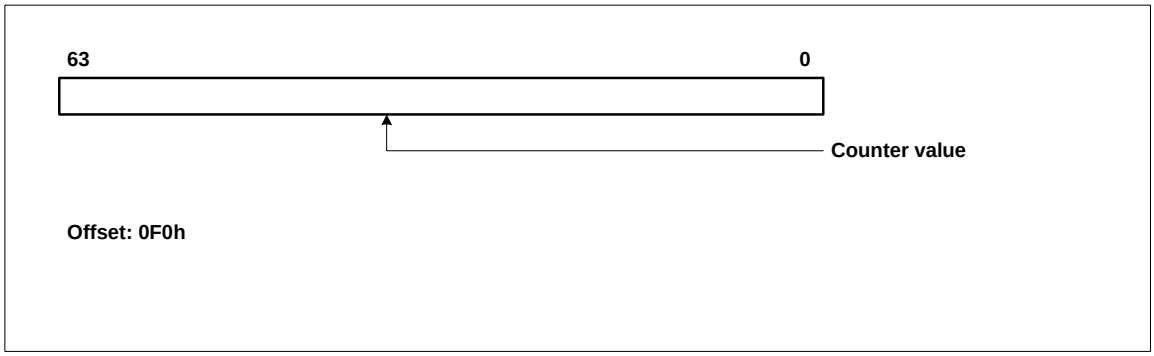


Figure 6 Main Counter Register

Main Counter Register Addressing

Offset	0F0h
Attribute	Read/Write
Size	64-bits
General Behavioral Rules	1. Software can access the various bytes in this register using 32-bit or 64-bit accesses. 32-bit accesses can be done to offset 0F0h or 0F4h. 64-bit accesses can be done to 0F0h. 32-bit accesses must not be done starting at: 0F1h, 0F2h, 0F3h, 0F5h, 0F6h, or 0F7h. 2. Writes to this register should only be done while the counter is halted. 3. Reads to this register return the current value of the main counter. 4. 32-bit counters will always return 0 for the upper 32-bits of this register. 5. If 32-bit software attempts to read a 64-bit counter, it should first halt the counter. Since this will delay the interrupts for all of the timers, this should be done only if the consequences are understood. It is strongly recommended that 32-bit software only operate the timer in 32-bit mode. Note: In a 64-bit platform, a 64-bit software attempts to read this 64-bit counter, it may still need to halt the counter and performs two 32-bit reads. It's due to the bus width of data path for accessing the main counter may only be 32-bit wide. 6. Reads to this register are monotonic. No two consecutive reads will return the same value. The second of two reads will always return a larger value (unless the timer has rolled over to 0).

Main Counter Register Field Definitions

Bit	Field Name	Description
63:0	**MAIN_COUNTER_VAL**	Bits 63:0 of the counter.

2.3.8 Timer N Configuration and Capabilities Register

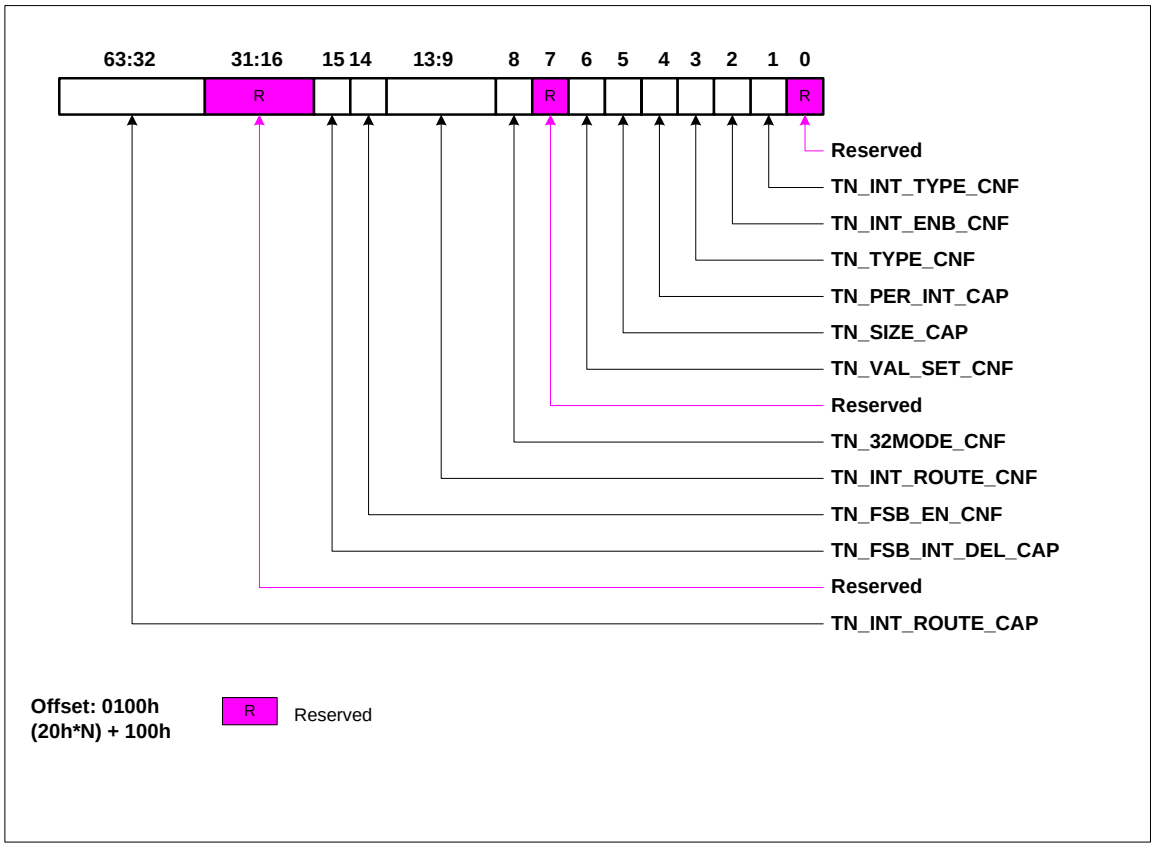
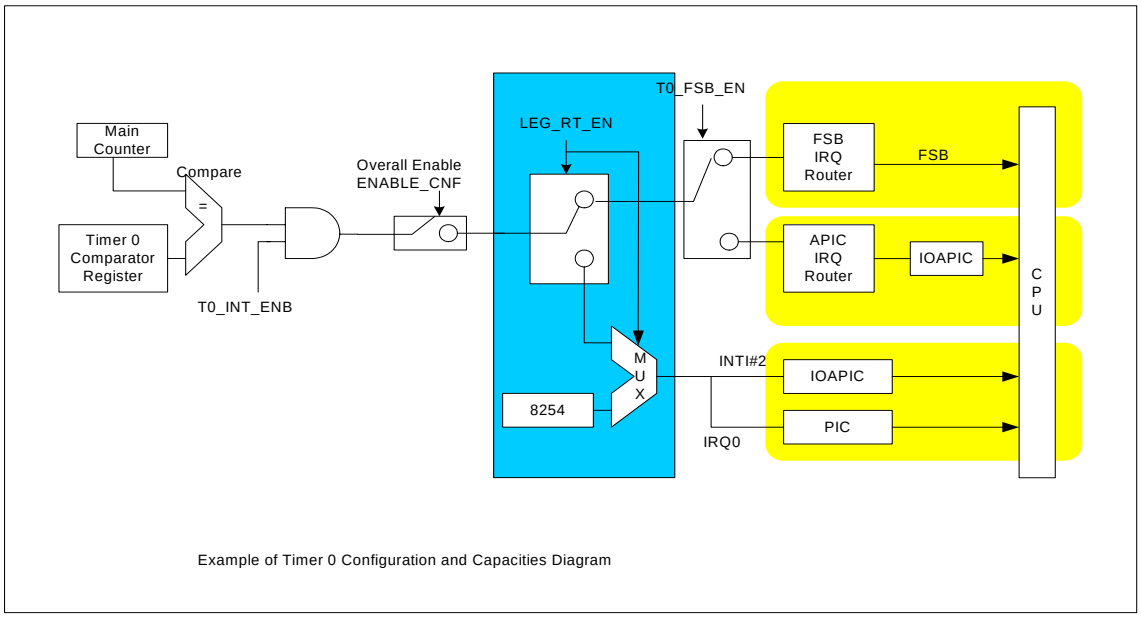


Figure 7 Timer N Configuration and Capability Register



Example of Timer 0 Configuration and Capacities Diagram

Timer N Configuration And Capability Register Addressing

Offset	Timer 0: 100h – 107h, Timer 1: 120h – 127h, Timer 2: 140h – 147h Timer n: (20h * n) + 100h - (20h * n) + 107h
Attribute	Read-Write
Size	64-bits
General Behavioral Rules	1. Software can access the various bytes in this register using 32-bit or 64-bit accesses. 32-bit accesses can be done to offset 1x0h or 1x4h. 64-bit accesses can be done to 1x0h. 32-bit accesses must not be done to 1x1h, 1x2h, 1x3h, 1x5h, 1x6h, 1x7h. 2. Reads or writes to unimplemented timers should not be attempted.

Timer N Configuration and Capability Register Field Definitions

Bit	Field Name	Description
64:32	**Tn_INT_ROUTE_CAP**	This 32-bit read-only field indicates to which interrupts in the I/O (x) APIC this timer's interrupt can be routed. This is used in conjunction with the **Tn_INT_ROUTE_CNF** field. Each bit in this field corresponds to a particular interrupt. For example, if this timer's interrupt can be mapped to interrupts 16, 18, 20, 22, or 24, then bits 16, 18, 20, 22, and 24 in this field will be set to 1. All other bits will be 0.
31:16	**Reserved**	In order to preserve usage of these bits in the future, software should always write a 0 to these bits until they are defined.
15	**Tn_FSB_INT_ DEL_CAP**	**FSB Interrupt Delivery:** (where n is the timer number: 00 to 31). If this read-only bit is 1, then the hardware supports a direct front-side bus delivery of this timer's interrupt.
14	**Tn_FSB_EN_CNF**	(where n is the timer number: 00 to 31). If the **Tn_FSB_INT_DEL_CAP** bit is set for this timer, then the software can set the **Tn_FSB_EN_CNF** bit to force the interrupts to be delivered directly as FSB messages, rather than using the I/O (x) APIC. In this case, the **Tn_INT_ROUTE_CNF** field in this register will be ignored. The **Tn_FSB_ROUTE** register will be used instead.
13:9	**Tn_INT_ROUTE_CNF**	**Interrupt Route:** (where n is the timer number: 00 to 31). This 5-bit field indicates the routing for the interrupt to the I/O APIC. A maximum value of 32 interrupts are supported. Default is 00h Software writes to this field to select which interrupt in the I/O (x) will be used for this timer's interrupt. If the value is not supported by this prarticular timer, then the value read back will not match what is written. The software must only write valid values. Note: If the LegacyReplacement Route bit is set, then Timers 0 and 1 will have a different routing, and this bit field has no effect for those two timers. Note: If the **Tn_FSB_INT_DEL_CN**F bit is set, then the interrupt will be delivered directly to the FSB, and this bit field has no effect.
8	**Tn_32MODE_CNF**	**Timer n 32-bit Mode:** (where n is the timer number: 00 to 31). Software can set this bit to force a 64-bit timer to behave as a 32-bit timer. This is typically needed if the software is not willing to halt the main counter to read or write a particular timer, and the software is not capable of doing an atomic 64-bit read to the timer. If the timer is not 64 bits wide, then this bit will always be read as 0 and writes will have no effect.
7	**Reserved**	In order to preserve usage of these bits in the future, software should always write a 0 to these bits until they are defined.
6	**Tn_VAL_SET_CNF**	**Timer n Value Set:** Software uses this bit only for timers that have been set to periodic mode. By writing this bit to a 1, the software is then allowed to directly set a periodic timer's accumulator. Software does NOT have to write this bit back to 0 (it automatically clears). Software should not write a 1 to this bit position if the timer is set to non-periodic mode.
5	**Tn_SIZE_CAP**	**Timer n Size:** (where n is the timer number: 00 to 31). This read-only field indicates the size of the timer. 1 = 64-bits, 0 = 32-bits.

4	**Tn_PER_INT_CAP**	**Periodic Interrupt Capable:** (where n is the timer number: 00 to 31). If this read-only bit is 1, then the hardware supports a periodic mode for this timer's interrupt.
3	**Tn_TYPE_CNF**	**Timer n Type:** (where n is the timer number: 00 to 31). If the corresponding **Tn_PER_INT_CAP (bit <4>)** bit is 0, then this bit will always return 0 when reads and writes will have no impact. If the corresponding **Tn_PER_INT_CAP (bit <4>)** bit is 1, then this bit is read/write, and can be used to enable the timer to generate a periodic interrupt. • Writing a 1 to this bit enables the timer to generate a periodic interrupt. • Writing a 0 to this bit enables the timer to generate a non-periodic interrupt.
2	**Tn_INT_ENB_CNF**	**Timer n Interrupt Enable:** (where n is the timer number: 00 to 31). This bit must be set to enable timer n to cause an interrupt when it times out. **Note:** If this bit is 0, the timer will still operate and generate appropriate status bits, but will not cause an interrupt.
1	**Tn_INT_TYPE_CNF**	**Timer n Interrupt Type:** (where n is the timer number: 00 to 31) • 0 = The timer interrupt is edge triggered. This means that an edge-type interrupt is generated. If another interrupt occurs, another edge will be generated. • 1 = The timer interrupt is level triggered. This means that a level-triggered interrupt is generated. The interrupt will be held active until it is cleared by writing to the bit in the General Interrupt Status Register. If another interrupt occurs before the interrupt is cleared, the interrupt will remain active.
0	**Reserved**	In order to preserve usage of these bits in the future, software should always write a 0 to these bits until they are defined.

2.3.9 Timer N Comparator Register

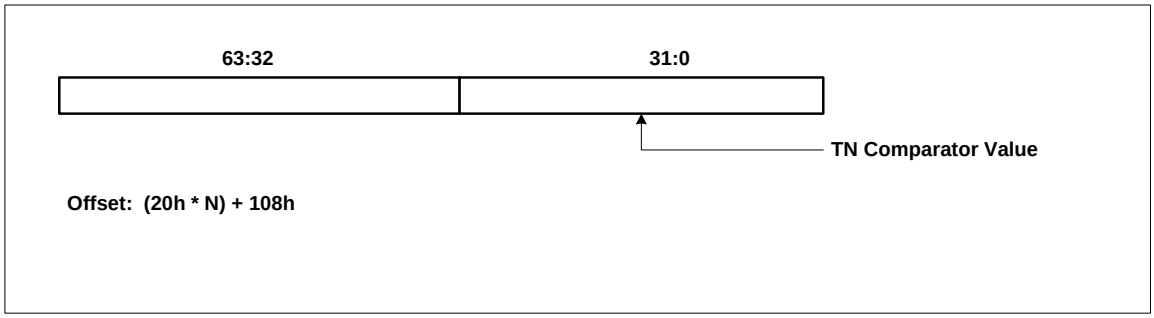
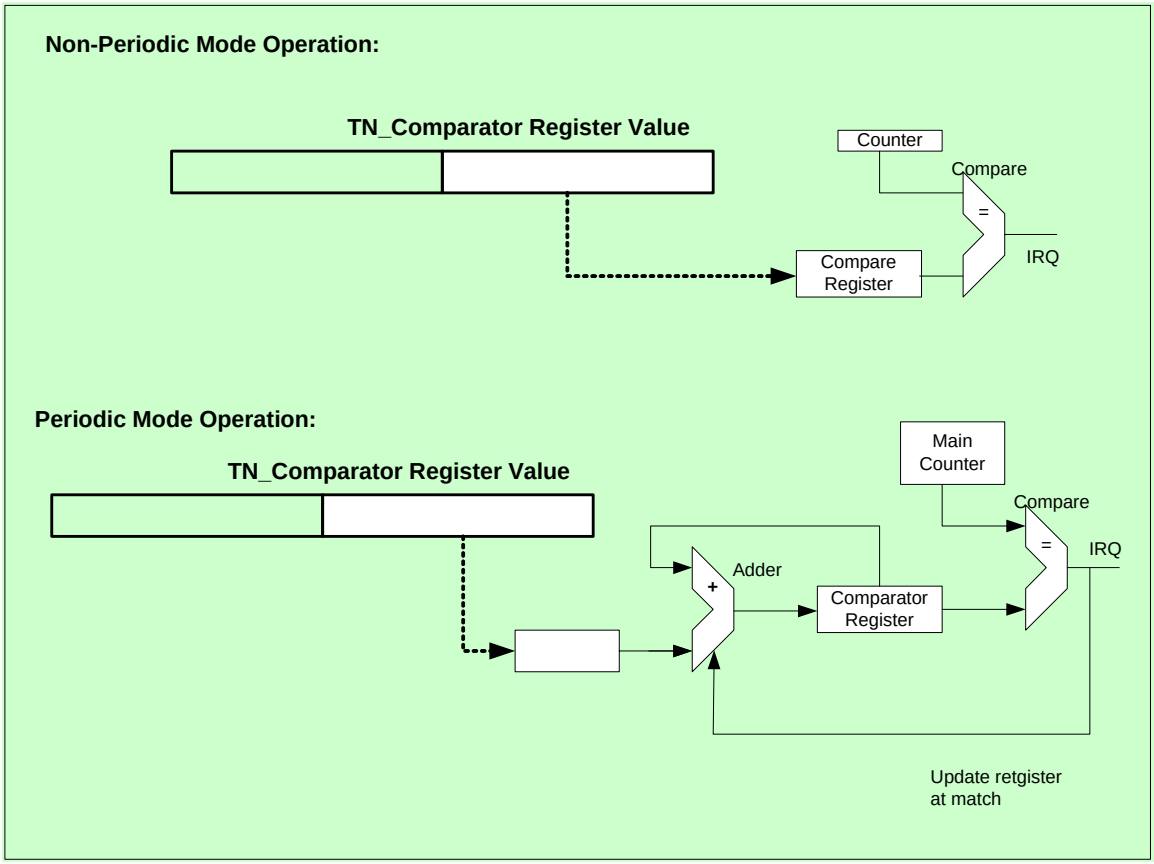


Figure 8 Timer N Comparator Register



2.3.9.1 Register Definition and Usage Model

Timer N Comparator Register Addressing

Offset	Timer 0: 108h – 10Fh Timer 1: 128h – 12Fh Timer 2: 148h – 14Fh Timer n: (20h * n) + 108h - (20h * n) + 10Fh
Attribute	Read-Write
Size	64-bits
General Behavioral Rules	1. Software can access the various bytes in this register using 32-bit or 64-bit accesses. 32-bit accesses can be done to offset 1x8h or 1xCh. 64-bit accesses can be done to 1x8h. 32-bit accesses must not be done to 1x9h, 1xAh, 1xBh, 1xDh, 1xEh, or 1xFh. 2. Reads to this register return the current value of the comparator. 3. If the timer is configured to non-periodic mode: • Writes to this register load the value against which the main counter should be compared for this timer. • When the main counter equals the value last written to this register, the corresponding interrupt can be generated (if so enabled). • The value in this register does not change based on the interrupt being generated. 4. If the timer is configured to periodic mode: • When the main counter equals the value last written to this register, the corresponding interrupt can be generated (if so enabled). • After the main counter equals the value in this register, the value in this register is increased by the value last written to the register. For example, if the value written to the register is 00000123h, then 1. An interrupt will be generated when the main counter reaches 00000123h. 2. The value in this register will then be adjusted by the hardware to 00000246h. 3. Another interrupt will be generated when the main counter reaches 00000246h 4. The value in this register will then be adjusted by the hardware to 00000369h • As each periodic interrupt occurs, the value in this register will increment. When the incremented value is greater than the maximum value possible for this register (FFFFFFFFh for a 32-bit timer or FFFFFFFFFFFFFFFFh for a 64-bit timer), the value will wrap around through 0. For example, if the current value in a 32-bit timer is FFFF0000h and the last value written to this register is 20000, then after the next interrupt the value will change to 00010000h 5. Default value for each timer is all 1's for the bits that are implemented. For example, a 32-bit timer will have a default value of 00000000FFFFFFFFh. A 64-bit timer will have a default value of FFFFFFFFFFFFFFFFh.

2.3.9.2 Periodic vs. Non-Periodic Modes

2.3.9.2.1 Non-Periodic Mode

This mode change can be thought of as creating a one-shot.

When a timer is set up for non-periodic mode, it will generate a value in the main counter that matches the value in the timer's comparator register. If the timer is set up for 32-bit mode, then it will generate another interrupt when the main counter wraps around.

During run-time, the value in the timer's comparator value register will not be changed by the hardware. Software can of course change the value.

WARNING: Software developers must be careful when programming the comparator registers. If the value written to the register is not sufficiently set far enough ahead of the current register value, then the counter may pass the value before it reaches the register and the interrupt will be missed.

Every timer is required to support the non-periodic mode of operation.

2.3.9.2.2 Periodic Mode

When a timer is set up for periodic mode, the software writes a value in the timer's comparator value register. When the main counter value matches the value in the timer's comparator value register, an interrupt can be generated. The hardware will then automatically increase the value in the comparator value register by the last value written to that register.

To make the periodic mode work properly, the main counter is typically written with a value of 0 so that the first interrupt occurs at the right point for the comparator. If the main counter is not set to 0, interrupts may not occur as expected.

During run-time, the value in the timer's comparator value register can be read by software to find out when the next periodic interrupt will be generated (not the rate at which it generates interrupts). Software is expected to retain the last value written to the comparator's value register (the rate at which interrupts are generated).

If software wants to change the periodic rate, it should write a new value to the comparator value register. At the point when the timer's comparator indicates a match, this new value will be added to derive the next matching point. So as to avoid race conditions where the new value is written just as a match occurs, either the main counter should be halted or the comparator disabled when the new periodic rate is written.

If the software resets the main counter, the value in the comparator's value register needs to reset as well. This can be done by setting the **Tn_VAL_SET_CNF** bit. Again, to avoid race conditions, this should be done with the main counter halted. The following usage model is expected:

1) Software clears the GLOBAL_ENABLE_CNF bit to prevent any interrupts
2) Software Clears the main counter by writing a value of 00000000h to it.
3) Software sets the TIMER0_VAL_SET_CNF bit.
4) Software writes the new value in the TIMER0_COMPARATOR_VAL register
5) Software sets the GLOBAL_ENABLE_CNF bit to enable interrupts.

2.3.9.2.3 Read/Write Paths for Periodic Mode Vs One-Shot Mode

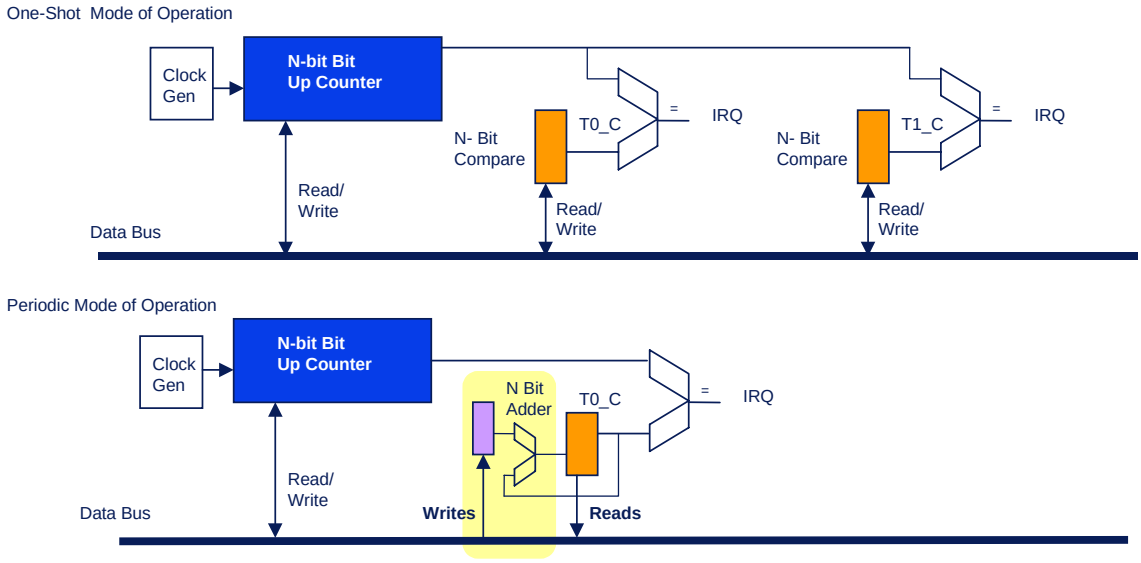


Figure 9 Read/Write paths for One-shot Vs Periodic modes of operation

For Event Timers that provide hardware support (i.e. the adder logic) for periodic mode of operation, the Timer N Comparator Register is overloaded as shown in Figure 9. In the periodic mode of operation, writes to this register will program the periodic interval value to be added to the contents of the match register at the next interrupt. Reads from this register will return the current contents of the match register at which the next interrupt will occur.

If it is necessary to save/restore the context of a periodic timer through ACPI Sleep state transitions, system software is expected to 'remember' the last value written to this register.

2.3.10 Timer N FSB Interrupt Route Register

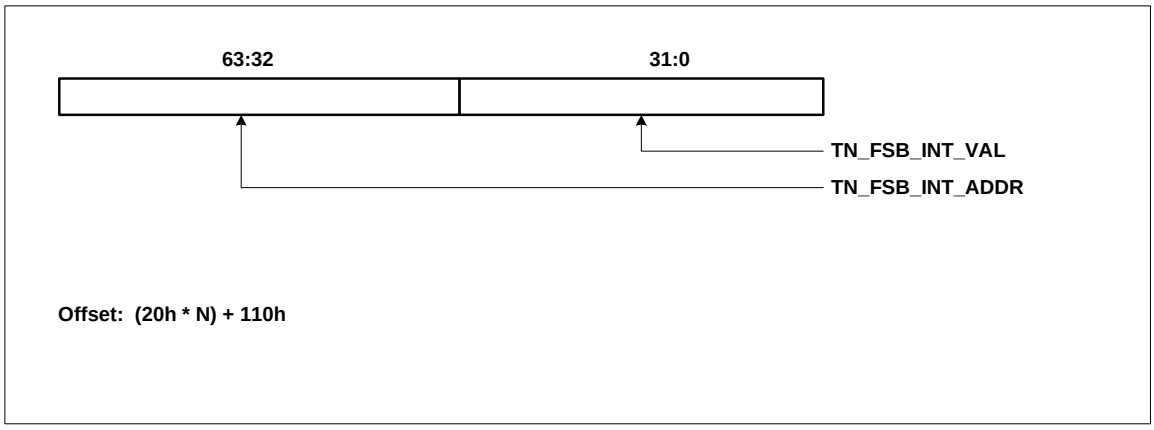


Figure 10 Timer N FSB Interrupt Route Register

Offset	Timer 0: 110h – 117h, Timer 1: 130h – 137h, Timer 2: 150h – 157h Timer n: (20h * n) + 110h - (20h * n) + 117h
Attribute	Read-Write
Size	64-bits
General Behavioral Rules	1. Software can access the various bytes in this register using 32-bit or 64-bit accesses. 32-bit accesses can be done to offset 1x0h or 1x4h. 64-bit accesses can be done to 1x0h. 32-bit accesses must not be done to 1x1h, 1x2h, 1x3h, 1x5h, 1x6h, 1x7h. 2. Reads or writes to unimplemented timers should not be attempted.

Bit	Field Name	Description
64:32	**Tn_FSB_INT_ADDR**	Software sets this 32-bit field to indicate the location that the FSB interrupt message should be written to.
31:0	**Tn_FSB_INT_VAL**	Software sets this 32-bit field to indicate that value that is written during the FSB interrupt message

2.4 Theory Of Operation

2.4.1 Timer Accuracy Rules

1. The timers are expected to be accurate over any 1 ms period to within 0.005% of the time specified in the timer resolution fields.
2. Within any 100-microsecond period, the timer is permitted to report a time that is up to 2 ticks too early or too late. Each tick must be less than or equal to 100 ns, so this represents an error of less than 0.2%.
3. The main counter must be an up-counter. 2 consecutive reads to the main counter may return the same value if the access latency to the timer is less than the clock period that feeds it. For back-back reads, the 2nd read must never return a value that is less than the 1st read, unless the counter has rolled over and actually reached the same value.

2.4.2 Interrupt Mapping

The interrupts associated with the various timers have several interrupt mapping options.

2.4.2.1 Mapping Option #1: LegacyReplacement Option

In this case, the LegacyReplacement Route bit (LEG_RT_CNF) will be set. This will force the following mapping:

Timer	8259 Mapping	APIC Mapping	Comment
0	IRQ0	IRQ2	In this case, the 8254 timer will not cause any interrupts
1	IRQ8	IRQ8	In this case, the RTC will not cause any interrupts.
2	As per IRQ Routing Field	As per IRQ Routing Field	

2.4.2.2 Mapping Option #2: Standard Option

In this case, the LegacyReplacement Route bit (LEG_RT_CNF) will be 0. Each timer has its own routing control. The interrupts can be routed to various interrupts in the I/O APIC. A capability field indicates which interrupts are valid options for the routing.

If a timer is set for edge-triggered mode, the timers should not be shared with any PCI interrupts.

2.4.2.3 Mapping Option #3: FSB Option

In this case, the interrupts are mapped directly to the FSB interrupts without going to the 8259 or I/O (x) APIC. To use this mode, the interrupt must be configured to edge-triggered mode. A separate configure bit must be set to enable this mode.

When the interrupt is delivered to the FSB, the message is delivered to the address indicated in the **Tn_FSB_INT_ADDR** field. The data value for the write cycle is specified in the **Tn_FSB_INT_VAL** field.

Notes:

- The FSB interrupt deliver option has HIGHER priority and is mutually exclusive to the standard interrupt delivery option. Thus, if the **Tn_FSB_EN_CNF** bit is set, the interrupts will be delivered via the FSB, rather than via the APIC or 8259.
- The FSB interrupt delivery can be used even when the LegacyReplacement mapping is used.

2.4.3 Periodic vs. Non-Periodic Modes

2.4.3.1 Non-Periodic Mode

This mode of operation provides a one-shot timer.

When a timer is set for non-periodic mode, it will generate an interrupt when the value in the main counter matches the value in the timer's comparator register. If the timer is set up for 32-bit mode, then it will generate another interrupt when the main counter wraps around.

During run-time, the value in the timer's comparator value register will not be changed by the hardware. Software can of course change the value.

WARNING:

Software developers must be careful when programming the comparator registers. If the value written to the register is not sufficiently set far enough ahead of the current register value, then the counter may pass the value before it reaches the register and the interrupt will be missed.

Every timer is required to support the non-periodic mode of operation.

2.4.3.2 Periodic Mode

When a timer is set for periodic mode, the software writes a value in the timer's comparator register. When the main counter value matches the value in the timer's comparator register, an interrupt can be generated. The hardware will then automatically increase the value in the compare register by the last value written to that register.

To make the periodic mode work properly, the main counter is typically written with a value of 0 so that the first interrupt occurs at the right point for the comparator. If the main counter is not set to 0, interrupts may not occur as expected.

During run-time, the value in the timer's comparator value register can be read by software to find out when the next periodic interrupt will be generated.

Software can also write to the comparator's match register to select a different period. A write to the register will not immediately be loaded into the comparator. It will only be added at the time the comparator triggers.

There is no mechanism to immediately change the periodic rate.

Each timer is NOT required to support this mode of operation. A capabilities bit indicates if the particular timer supports periodic mode. The reason for this is that supporting the periodic mode adds a significant amount of gates.

2.4.4 Enabling the Timers

The BIOS or OS PnP code should:

1. Assign Base Address to each Timer Block in the System.
2. Route the interrupts. This includes the LegacyReplacement Route bit, Interrupt Route bit (for each timer), interrupt type (to select the edge or level type for each timer).
3. If system BIOS is used to enable the timers, then report memory and interrupt resources consumed by each Timer Block to OS Configuration Manager using HPET.

The Device Driver code should do the following for an available timer:

1. Set the timer type field (selects one-shot or periodic).
2. Set the interrupt enable
3. Set the comparator match
4. Set the Overall Enable bit (Offset 04h, bit 0). This starts the main counter and enables comparators to deliver interrupts.

2.4.5 Interrupt Levels

The interrupts are all active high.

If the interrupts are mapped to the I/O APIC and set for level-triggered mode, they can be shared with PCI interrupts.

2.4.6 Handling Interrupts

If each timer has a unique interrupt and the timer has been configured for edge-triggered mode, then there are no specific steps required. No read is required to process the interrupt.

If a timer has been configured to level-triggered mode, then its interrupt must be cleared by the software. This is done by reading the interrupt status register and writing a 1 back to the bit position for the interrupt to be cleared.

Independent of the mode, software can read the value in the main counter to see how much time has passed from when the interrupt was generated and when it was first serviced.

If a timer is set up to generate a periodic interrupt, the software can check to see how much time remains until the next interrupt by reading the main counter.

2.4.7 Issues related to 64-bit Timers with 32-bit CPUs

A 32-bit timer can be read directly using processors that are capable of 32-bit or 64-bit instructions. However, a 32-bit processor may not be able to directly read a 64-bit timer. A race condition comes up if a 32-bit CPU reads the 64-bit register using two separate 32-bit reads. An accuracy problem may be arise if just after reading one half, the other half rolls over and changes the first half.

If a 32-bit CPU needs to access a 64-bit register value, it must first halt the timer before reading both the upper and lower 32-bits of the timer.

If a 32-bit CPU does not want to halt the timer, it can use the 64-bit timer as a 32-bit timer by setting the TIMERn_32MODE_CNF bit. This will cause the timer to behave as a 32-bit timer. The upper 32-bits will always be 0.

> Note: In a 64-bit platform, a 64-bit software attempts to read this 64-bit counter, it may still need to halt the counter and performs two 32-bit reads. It's due to the bus width of data path for accessing the main counter may only be 32-bit wide.

3. Enumeration & Configuration of HPET

Operating System software must discover/configure platform timers early in the OS boot process. Typically, the OS has to establish basic timer services before it can begin loading drivers. This requires enumeration of timer hardware to be handled by system BIOS tables (ie ACPI Tables) versus ACPI name space. System Resources consumed by the Event Timer hardware (memory and interrupts) should be reported by the System BIOS using ACPI Name space.

3.1 Initial State of Event Timer Hardware

- Main Counter is Halted and Zeroed
- Comparator Match Registers reset to all 1's.
- All interrupts are disabled
  - General Configuration and Capability Register [Offset 0x010]<1:0> = 00
    - Global IRQ Enable bit comes up disabled…no comparators can deliver interrupts
    - LegacyReplacement IRQ Routing Enable bit comes up disabled…8254 is on IRQ0, RTC is on IRQ8

3.2 BIOS Initialization

3.2.1 Assign memory to Timer Block(s)

Map each HPET (MMT) block to CPU memory space using implementation specific Base Address Registers. Each HPET block in the system will consume 1K of system memory.

3.2.2 HPET Block Interrupt Routing

Except for the case where HPET are being used to replace 8254/RTC functionality, all HPET interrupts are disabled. The OS is responsible for establishing interrupt routing/delivery metrics prior to utilizing any given comparator within a Timer Block.

3.2.2.1 Routing Interrupts for HPET Blocks that do not support 8254/RTC IRQ Routing

In general, system BIOS is not required to assign or report HPET interrupts in system name space. The Power-On-Default state of all compare interrupts should be disabled as shown in the following table.

Device	Interrupt Routing	Comments
8254	IRQ0, INTI2	8254 signals via PIC/APIC using IRQ0/INTI2 → LegacyReplacement IRQ Routing Disabled for Comparator_0
RTC	IRQ8, INTI8	RTC signals via PIC/APIC using IRQ8/INTI8 → LegacyReplacement IRQ Routing Disabled for Comparator_1
Compare 0	INTI [xx]	BIOS leaves Disabled → OS responsible for establishing irq routing prior to using this comparator
Compare 1	INTI [xx]	BIOS leaves Disabled → OS responsible for establishing irq metrics prior to using this comparator
Compare 2	INTI [xx]	BIOS leaves Disabled → OS responsible for establishing irq metrics prior to using this comparator
…	…	
Compare 31	INTI [xx]	BIOS leaves Disabled → OS responsible for establishing irq metrics prior to using this comparator

3.2.2.2 Routing Interrupts for HPET Blocks that support 8254/RTC IRQs

The exception to this rule is for the single HPET block in the system that may optionally support legacy 8254 & LegacyReplacement RTC irq routings for the compare interrupts. Assuming platform does not have 8254/RTC hardware or does not want to support this legacy timer hardware, for this case, System BIOS should set the LegacyReplacement Route bit and report IRQ0 & IRQ8 as being consumed by the HPET block in system name space:

Device	Interrupt Routing	Comments
8254	Not connected	BIOS sets LegacyReplacement Route bit (LEG_RT_CNF) > LegacyReplacement IRQ Routing Enabled for Comparator_0 • If present, 8254 will not cause any interrupts • If present, 8254 will still consume legacy i/o range
RTC	Not connected	BIOS sets LegacyReplacement Route bit (LEG_RT_CNF) > LegacyReplacement IRQ Routing Enabled for Comparator_1 • If present, RTC Periodic Interrupt Function will not cause any interrupts. • RTC Alarm function (still required) will signal interrupts via SCI • RTC CMOS function (still required) will consume i/o range
Compare 0	IRQ0, INTI2	BIOS sets LegacyReplacement Route bit (LEG_RT_CNF) • Comparator_0 replaces 8254 PIT Function
Compare 1	IRQ8, INTI8	BIOS sets LegacyReplacement Route bit (LEG_RT_CNF) • Comparator_0 replaces 8254 PIT Function
Compare 2	INTI [xx]	
…		
Compare 31	INTI [xx]	

Note 1: The use of LegacyReplacement IRQ Routing for C0 & C1 does not preclude delivery of IRQ0/IRQ8 via FSB.

3.2.3 Considerations for Platforms without Legacy Timers

If it is necessary to maintain DOS compatibility at INT 19 (or if necessary by OS to have periodic timer ticks running when hand-over occurs from BIOS to OS for IPL), BIOS can use Timer_0 in periodic mode (Vs 8254).

IA-PC HPET do not replace RTC Time of Day, RTC Alarm, and RTC CMOS functionality. IA-PC Multimedia Event Timer architecture supplements/replaces only the RTC Periodic Interrupt function. When the event timer is using IRQ8, RTC Alarm function will signal interrupts using SCI.

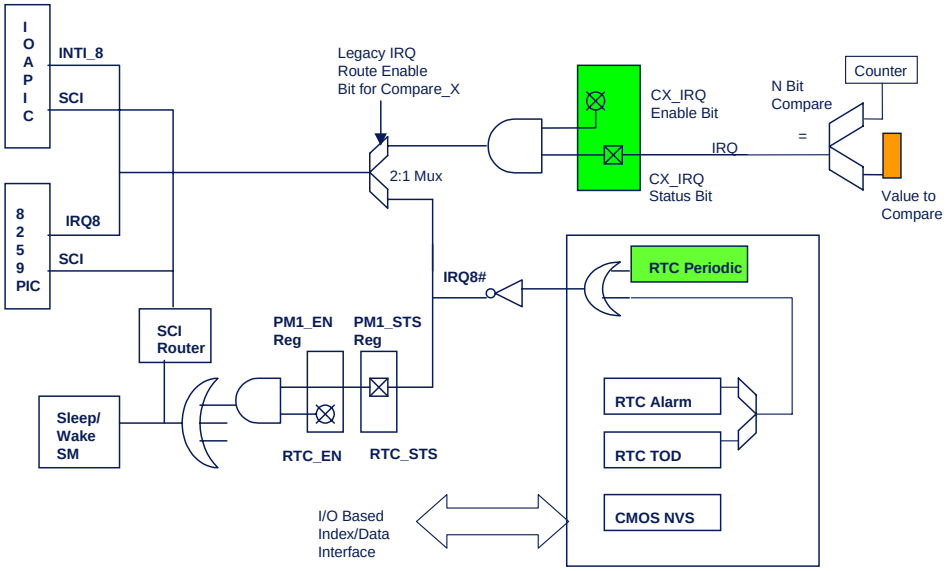


Figure 11 RTC Functionality replaced by IAPC HPET

The IA-PC HPET Architecture switches out both 8254 & RTC Periodic Interrupt Functions together.

For the platforms without Legacy Timers, System BIOS needs to mark the 8254 and RTC Periodic functions as ‘missing’ using the ACPI 2.0 proposed ‘legacy free’ flags.

3.2.4 Create ACPI 2.0 HPET Description Table (HPET)

HPET is a means to report the Base Addresses of each Event Timer Block early in the OS boot process. HPET is needed to allow Operating Systems to discover event timers and establish basic timer services for driver load.

Table 3 HPET Description Table

Field	Byte Length decimal	Byte Offset decimal	Description
Header			
Signature	4	0	'HPET'. Signature for the Event Timer Description Table.
Length	4	4	Length, in bytes, of the entire Event Timer Description Table.
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID.
OEM Table ID	8	16	For the Event Timer Description Table, the table ID is the manufacturer model ID.
OEM Revision	4	24	OEM revision of Event Timer Description Table for supplied OEM Table ID.
Creator ID	4	28	Vendor ID of utility that created the table. For the DSDT, RSDT, SSDT, and PSDT tables, this is the ID for the ASL Compiler.
Creator Revision	4	32	Revision of utility that created the table. For the DSDT, RSDT, SSDT, and PSDT tables, this is the revision for the ASL Compiler.
Event Timer Block ID *Note #1	4	36	Hardware ID of Event Timer Block: Contents of General_Cap&ID Reg of Timer Block [31:16] = PCI Vendor ID of 1st Timer Block [15] = LegacyReplacement IRQ Routing Capable [14] = Reserved **[13] = COUNT_SIZE_CAP counter size** [12:8] = Number of Comparators in 1st Timer Block [7:0] = Hardware Rev ID
BASE_ADDRESS Lower 32-bit *Note #2	12	40	The lower 32-bit base address of Event Timer Block. Each Event Timer Block consumes 1K of system memory, regardless of how many comparators are actually implemented by the hardware.
HPET Number	**1**	**52**	**This one byte field indicates the HPET sequence number. 0 = 1st table, 1 = 2nd table and so forth. This field is written by BIOS at boot time and should not be altered by any other software.**
Main Counter Minimum Clock_tick in Periodic Mode *Note#3	**2**	**53**	**Unit: Clock tick The minimum clock ticks can be set without lost interrupts while the counter is programmed to operate in periodic mode**

Field	Byte Length decimal	Byte Offset decimal	Description
Page Protection And OEM Attribute	**1**	**55**	The lower 4-bit (bit <0..3>) of this field describes the timer hardware capability to guarantee the page protection. This information is required for the OSes that want to expose this timer to the user space: 0 = no guarantee for page protection. 1 = 4KB page protected, access to the adjacent 3KB space will not generate machine check or compromise the system security. 2 = 64KB page protected, access to the adjacent 63KB space will not generate machine check or compromise the system security. 3~ 15 = Reserved for future use. **The upper 4-bit (bits <4..7> of this field is reserved for OEM attributes:** OEM can use this field for its implementation specific attributes.

*Note #1: This field provides a quick access by devices those need to know the HPET implementation.

*Note #2: This is a 12-byte ACPI address format:

GAS (Generic Address Structure) -- ACPI Address Format:

Byte #1 – Address_Space_ID : 0 – System Memory
1 – System I/O
Byte #2 – Register_Bit_Width
Byte #3 – Register_Bit_Offset
Byte #4 – Reserved
Byte #5 to 12 – 64-bit of address

* Note #3: This field is written by BIOS and may be chipset and/or platform dependent. This indicates the minimum value that must be used for any counter programmed in periodic mode to avoid lost interrupts. For any counter x that has been configured for periodic mode, the number can be programmed in any Tx_Compare Register must be greater than P, where

P = (Minimum Period) / (Main counter period) in order to avoid lost interrupts.

For the case where there may be additional Event Timer Blocks implemented in the system, their base address's would be described in ACPI Name space.

Only 1 Event Timer Block needs to be described in the HPET in order to boot strap the OS.

For "legacy free" platforms that do not implement the 8254 Timer/RTC Periodic Interrupt logic, the Event Timer Block described in the HPET would be the one that provides functionality to replace the 8254/RTC Periodic Interrupt Logic.

Object Description

_HID	Named object that provides the interface's Plug and Play identifier. This value is set to PNP0103. _HID is a standard device configuration control method defined in ACPI 2.0 Spec section 6.1.4, "_HID (Hardware ID)."
_STR	Named object (optional) that evaluates to a Unicode string that may be used by an OS to provide information to an end user describing the device. __STR is a standard device configuration control method defined in ACPI 2.0 Spec section 6.1.5, "_STR (String)."
_CRS	Named object that returns the Event Timer's current resource settings. Event Timer is considered static resources; hence only return its defined resources. _CRS is a standard device configuration control method defined in ACPI 2.0 Spec section 6.2.1, "_CRS (Current Resource Settings)."
PAGE	Object that specifies the page protection capability, as defined in the HPET Description Table.
ATTR	Object that specifies the timer attributes, as defined in the HPET Description Table

3.2.5 Describe Event Timer(s) in ACPI Name space

System BIOS must report memory and interrupt resources consumed by each Event Timer block in ACPI Name space.

Event Timer(s) memory assignments are established by the system BIOS on per Timer Block basis. Event Timer(s) memory assignments are reported to the OS by the BIOS using HPET Table and in ACPI Name space.

Event Timer IRQ assignments need only be established for the single Timer Block that may optionally provide hardware support to supplement/replace legacy 8254 and legacy RTC hardware with these new HPET. For this case, system bios reports interrupt resources consumed by the Timer Block in ACPI Name space. For the case where the Timer Block does not provide legacy 8254/legacy RTC hardware replacement, system bios is not required to establish or report Event Timer Interrupt assignments. The OS is expected to assign event timer interrupts prior to utilizing any given comparator in the Timer Block.

PNP0103 is Microsoft assigned generic PNPID for IA-PC HPET blocks.

3.2.5.1 ACPI Name Space Example

```
Device (HPETBlock)
    _HID         PNP0103                    // newly assigned PNPID for IAPC HPET
    _UID         0                          // Optional : used if there are more than 1 timer blocks
    _CRS    (    memory range consumed      // Report 1K of memory consumed by this Timer Block
                 IRQs consumed         )    // Optional :  only used if BIOS allocates Interrupts[1]
```

Notes:

1. For case where Timer Block is configured to consume IRQ0/IRQ8 AND Legacy 8254/Legacy RTC hardware still exists, the device objects associated with 8254 & RTC devices should not report IRQ0/IRQ8 as "consumed resources."

3.2.6 Recommendations for OS Initialization code

From ACPI HPET table, the OS initialization code extracts bios-assigned base address for each timer block implemented in the system. The OS initialization code can directly query the timer hardware to discover detailed attributes and configurations supported by the timer hardware implementation.

IA-PC HPET architecture has configuration and capability reporting mechanisms designed in at the register level. OS initialization code can query these registers to determine, establish or override (for example: default IRQ assignments established by the BIOS) Event Timer hardware configurations.

The following registers provide enumeration info to system software:

General Capabilities & ID Register Offset 00h

- Clock Frequency
- Width of Main Counter
- Vendor ID/Hardware ID
- LegacyReplacement Timer IRQ Routing Capable (or not)
- Number of Comparator's Implemented

To determine / override the default IRQ assignments established by the BIOS:

General Configuration Register Offset 04h

- Global Interrupt Enable Bit
- LegacyReplacement Timer IRQ Routing Enable Bit

Timer N Configuration & Capabilities Register

1 for each Comparator implemented:

Offsets:
Timer 0: 108h – 10Fh
Timer 1: 128h – 12Fh
Timer 2: 148h – 14Fh
Timer n: (20h * n) + 108h - (20h * n) + 10Fh

- Width of Comparator
  - Comparator Configuration bit which allows 64bit comparator to behave like a 32bit comparator.
- Per Comparator Interrupt Enable
  - Edge Vs Level selection
  - IOAPIC IRQ Routing capabilities reporting
  - IO APIC IRQ Router settings
  - FSB IRQ Delivery Capable (or not)
  - FSB IRQ Delivery Enable bit
- Operating Modes : Periodic Vs One-Shot

Timer_N_FSB_Route Register Offset: 1 for each Comparator Implemented

- Address to which the FSB interrupt message should be written.
- Data Value/Vector to be used with FSB interrupt message.